EDLBUILD – DISPLAY GENERATION FOR THE EPICS EDM DISPLAY MANAGER

R. Keitel
TRIUMF, 4004 Wesbrook Mall, Vancouver, B.C., Canada

ABSTRACT

"edm" is the most recent display editing and managing tool in the EPICS tool kit [1]. A Perl module library "EdlBuild" was developed, which allows programmed generation of edm screens and supplements the standard interactive screen development process. EdlBuild uses a syntax similar to Perl/Tk. It supports all edm widgets and allows configuration of all widget parameters. Easy configuration of site-specific "look-and-feel" is supported and hooks are provided for simple addition of site-specific widgets.

INTRODUCTION

The face of a modern accelerator control system is its graphical user interface (GUI). The EPICS toolkit, like most commercial control system toolkits, contains GUI-builder tools for interactively designing operator interface screens. All these EPICS display editing tools save display information in display data files, which are parsed and rendered by a display manager. The display editor and display manager may be separate tools, which is the case for the edd/dm combination used at the ISAC radioactive beam facility. Newer EPICS tools, such as medm, dm2k, and their latest sibling, edm combine both functions into a single utility.

With increasing size of a control system, there is an increasing demand for program-generated display data files. Interactive generation of certain types of screens becomes very repetitive, unproductive and error-prone. Unfortunately, none of the EPICS display editors/managers provide this functionality. Different laboratories implement different, usually partial, solutions to satisfy their particular needs.

At the ISAC control system, we use more than 600 device control panel screens with interlock information, which is extracted from our PLC programs [2]. As part of our quality assurance program, these panels must be re-generated whenever an interlock is changed. In addition, many screens are generated based on information extracted from our relational database and the EPICS run-time databases. As we prepare to abandon edd/dm in favour of edm, it is mandatory to adapt our display generating tools to support the edm data format.

THE EDLBUILD LIBRARY

The "EdlBuild" library presents a general solution for program-generated display data files. All currently existing edm widgets are supported, all widget properties can be configured, and future widgets are easily supported.

EdlBuild was implemented as a set of Perl modules using object oriented Perl. The EdlBuild Application Programming Interface (API) uses a syntax similar to the Perl/Tk module. EdlBuild makes use of the fact that all edm widget properties are name/value pairs.

A Perl script for generating an edm display data file contains the following steps:

1. a call to the screen constructor function ("new Edl( )"), which takes as parameter the desired display data file name and returns a handle to the new screen
2. for each widget on the screen, a call to the widget's generating function using the screen handle. The generating function takes as parameter an associative array of widget properties and returns a handle to the widget. The provided widget properties override the widget defaults.
3. optionally reconfigure any widget at any time with a call to its configure function using the widget handle. The configure function takes as parameter an associative array of widget properties.
4. a finish call using the screen handle, which writes the display data file.

Figure 1 shows an example Perl script, which generates a simple display. The rendering of the resulting display data file by edm is shown in Figure 2.

```
#!/usr/bin/perl -w

use strict;
use lib '/usr/local/perllib/';  # path to EdlBuild
use Edl;                         # EdlBuild module

# get handle to screen
my $screen = new Edl('test.edl');

# add a rectangle
$screen->rectangle(x => 10, y => 10, w => 100, h => 15,
      lineColor => 'index 10', fill => 1, fillColor => 'index 9');

# add a live text display
$screen->text_monitor(x => 10, y => 30, w => 100, h => 15,
      fgColor => 'index 22', controlPv => '"ITW:IG1:RDVAC"');

#reconfigure screen size
$screen->configure(w => 150, h => 150);

$screen->finish();
```

Fig. 1: EdlBuild script example

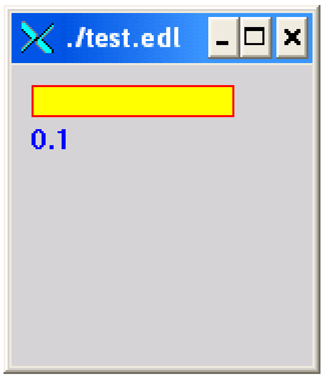

Figure 2: edm rendering of display data file generated by the example script

IMPLEMENTATION

Perl Modules

The EdlBuild library consists of

- a "main" module Edl.pm
- one module for each widget, such as EdlRectangle.pm, EdlTextMonitor.pm, etc.
- modules EdlDefault.pm, EdlSiteUse.pm, EdlSiteWidgets.pm for supporting site-specific requirements (see next section)

The main module Edl.pm includes all supported widget modules and contains the bulk of the code. It contains a generating function for each supported widget and all setup and configuration code. The generating function calls the widget constructor and links the new widget into internal lists.

Each widget function contains a widget property template in an associative array, a constructor function and a configuration function. Both these functions are wrappers, which pass parameters to functions of the main module. Therefore, the widget modules contain very little, but identical code and differ only in the property template data.

Configuring Site-Specific Look-and-Feel

The common look-and-feel of the generated displays can be modified by editing the default values of the widget property templates in the 30+ widget modules. A single modification point for commonly used colors and a default font is provided in the module EdlDefault.pm, which defines variables for these properties.

Adding Support for Site-SpecificWidgets

The edm display tool is extensible and EdlBuild makes it easy to add support for site-specific widgets. For each new widget, a widget module must be provided with the property template of the new widget. The new widget must then be registered by editing the modules EdlSiteUse.pm and EdlSiteWidgets.pm, which contain templates to help with this task.

CONCLUSION

The EdlBuild modules were successfully used to rebuild all automatically generated screens at ISAC for use with edm. In addition, the badlfish dm-to-edm-converter [3] was modified to use the EdlBuild API, which allowed to overcome some SLAC specific limitations in badlfish. The resulting Perl tool (Tadl2edl.pl) was used to convert all ISAC screens, which had been interactively generated by edd.

The switch from using edm/dm to edm at ISAC can now be planned for the near future.

REFERENCES

[1] John Sinclair, http://ics-web1.sns.ornl.gov/edm/

[2] R.Keitel, R. Nussbaumer, ‘Automated Checking and Visualization of Interlocks in the ISAC Control System’, ICALEPCS01, San Jose

[3] Dayle Kotturi, http://www.slac.stanford.edu/grp/cd/soft/epics/lcls/badlfish/